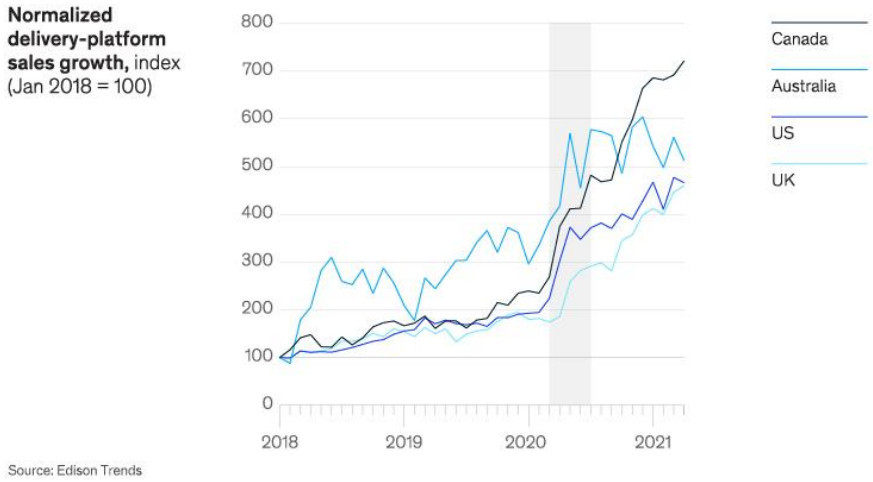
Report

Case Study

Last years we went through a pandemic and it has shown us how important delivery is in our daily lives which is why we want to make an app that helps people get the food they want right on their door steps.

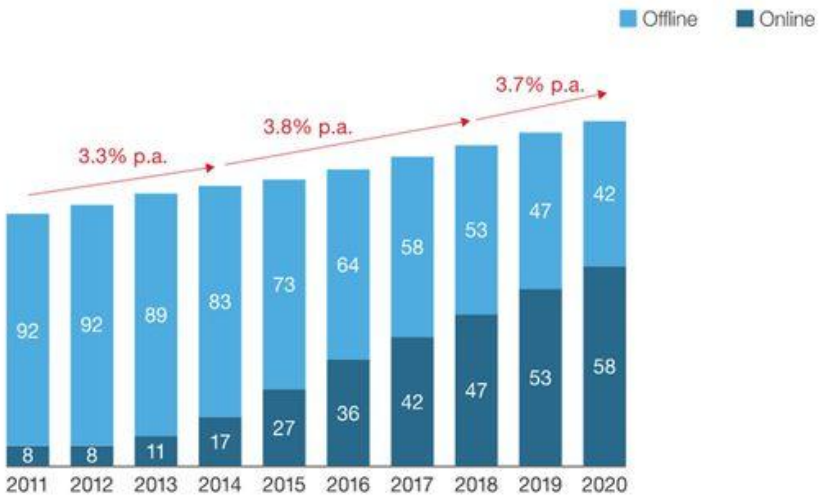
Since pandemic-related lockdowns started in March 2020, the growing food-delivery business has spiked to new heights in the most mature markets.

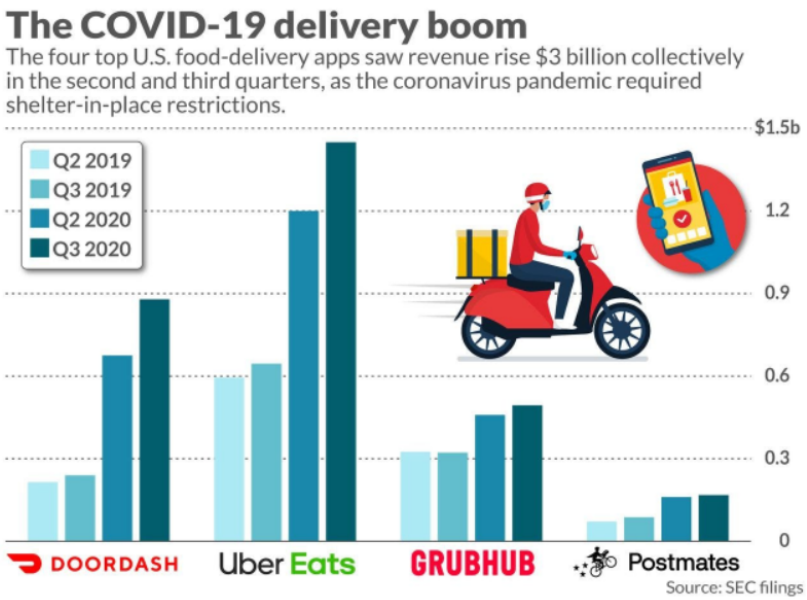


Source: Edison Trends

McKinsey & Company

The food-delivery market has the potential for robust growth.





Unlike the traditional delivery we offer new features such as: buying from different restaurants in the same order, statistics for users to see how the price and how the food orders evolved over time, and the app's own currency, which will make giving coupons and other awkward situations, like a canceled order, easier to handle. We believe these improvements will make the app more desirable and will encourage customers to explore more restaurants. How many times did you want to try food from another restaurant but weren't sure if the food would be good and went to the usual place? Or if you ate a pizza in a certain restaurant and it was delicious, this doesn't guarantee that the sushi will also taste good. Making various specific statistics accessible to the customers will be quite attractive to them. Another issue encountered when one wants to order something is the fact that you can't order from multiple restaurants at once, but if handled properly it is possible to be done so we can please a bigger variety of customers.

Technical details about existing solutions

We can't know for sure what the architecture or the technologies are used on a delivery app, but we can guess.

- Architecture

Because such apps are multiplatform, the backend server could be structured as an API and the frontend will be separate from it. Also the company can make use of cloud technologies such as queue storage to handle multiple requests, database, cloud storage and other ones.

- Technologies

Frontend: depending on the platform the app is running, the frontend can be developed in: Angular, React, Vue, Proton and other popular js frameworks.

Backend: could be written with different frameworks: express, django, spring, etc. Also, the app can use Paypal or Stripe for managing payments. The location is also required so a gps service will be used. Other services used will include: GPS tracking service when the user will see in real time the courier location, a service to get the delivery guy directions.

- Marketing Approaches

There are 2 business models that are the most popular:

- *Platform-to-Consumer Mode*l: used by companies like UberEats, this model brings together all of the offerings of small restaurants and allows customers to order from them via the app platform. Here, the restaurant partners are responsible for their own fleet of couriers.This model seems to be more profitable as platforms generate revenue from both customers and restaurants by charging a fixed amount as a fee for each other.[1]
- *Delivery Service Aggregators*: used by companies like GrubHub, in this model the platform works as a bridge between a customer (user) and numerous local restaurants in the aggregator model. In case customers face any problems with their deliveries or orders, they offer customer support on behalf of these restaurants. Here, the platform charges a fixed amount of fee on every successful transaction.Moreover, some of the companies have tried to adopt subscription models in which customers are liable to pay a monthly fee in order to leverage other benefits such as discounts and free home delivery.[1]

To “spread the word around” an usual approach is the use of ads on TV, social media and other sites. To attract customers they have various offers, such as: the first delivery is free, or you gain a 10 Lei coupon on your first order, or various discounts, from famous restaurants in particular (like McDonald’s, KFC etc).

Technical details

The app will be structured in two main components, the frontend, and the backend that will be like a service exposing an api.

Frontend: Angular with Typescript because it’s a popular component based framework

Backend: Express framework with Typescript because it’s very easy to write an API with it, unlike Spring Boot framework, and the Typescript programming language is strongly typed with the programmer choice to make it weakly type, so best of both worlds.

Cloud Services: Google Cloud will be used as a cloud provider. The services includes: postgres database because it is more suitable if the app grows to make more complex statistics than a document based database, cloud storage for keeping photos, container registry for keeping docker images, and cloud run to run the backend server.

Business canvas

Business Model Canvas

Key Partners

- Curierii.Ro
- Sinapseria
- Ciclocurier
- McDonalds
- KFC
- Spartan
- etc

Key Activities

- Fast & Safe delivery
- Platform maintenance
- Software development

Key Resources

- Software developers
- Software
- Cloud subscription
- Couriers

Value Propositions

Customer
- Easy to use
- Making every food desirable
- Fast & Safe delivery
- Convenience/Usability

Restaurant
- Market visibility

Customer Relationships

- Trust
- Availability

Channels

- Social networks(Ads)
- Email
- Flyers
- .ro tld

Customer Segments

- Customer
- Restaurant owner
- Mass-Market

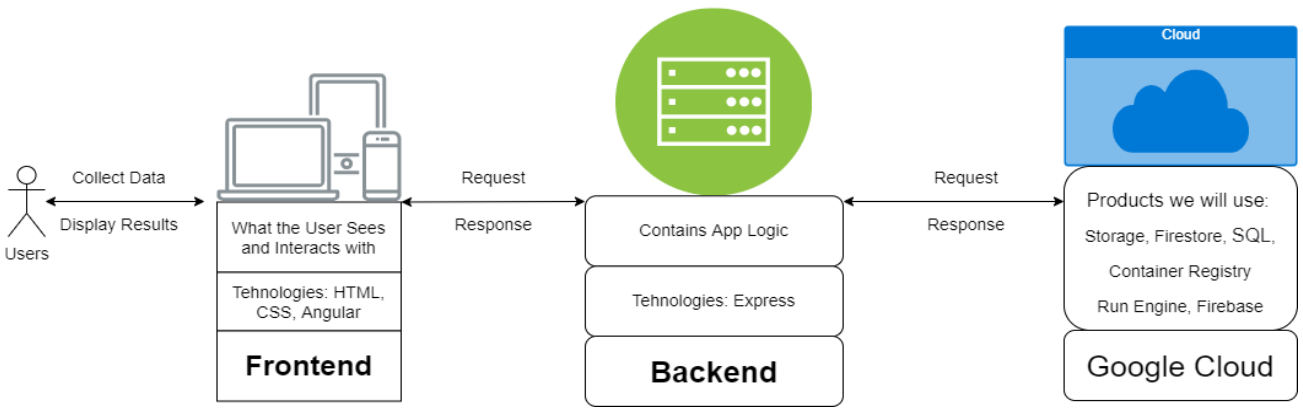
Cost Structure

- Software development
- Cloud subscription
- Marketing
- Business-Model: Platform to Consumer Model

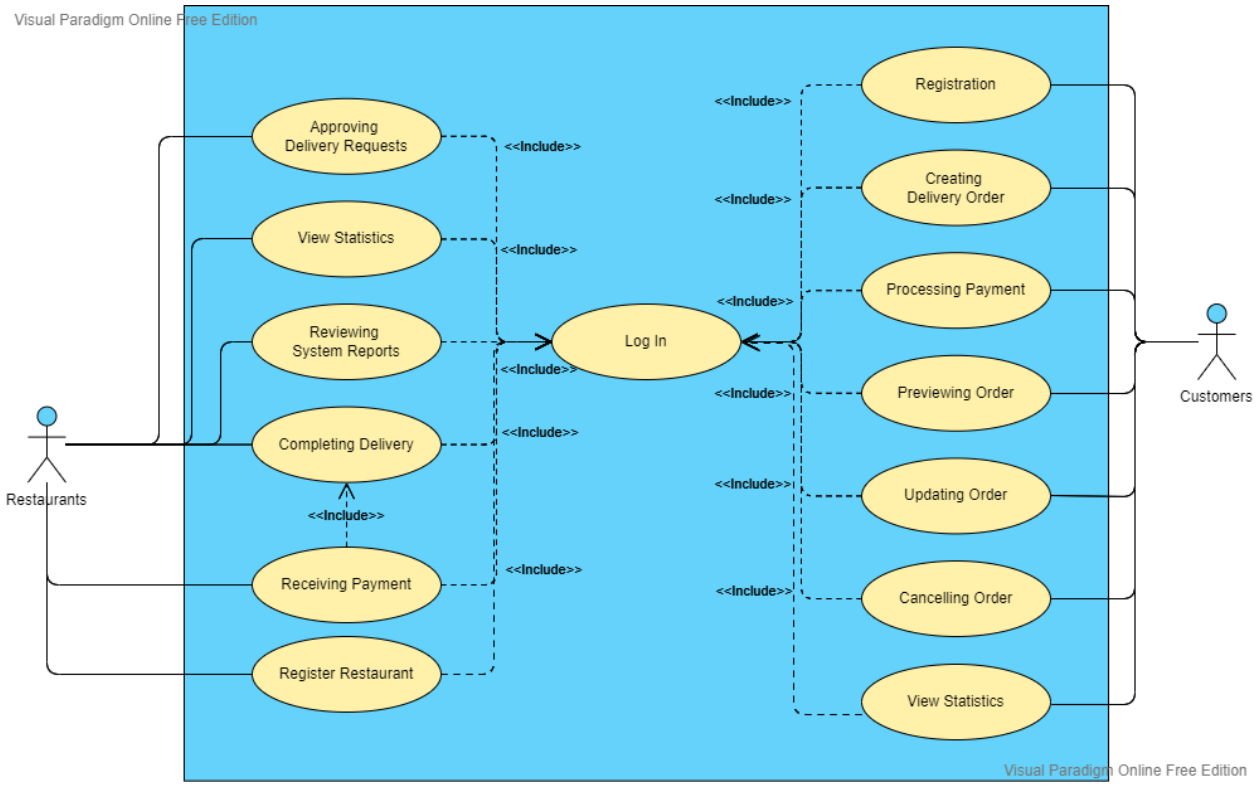
Revenue Streams

- A fee for both customers and restaurants
- Subscription
- Pay for visibility

Architectural diagram



Use-Case diagram



API

The api documentation can be found here:
https://app.swaggerhub.com/apis/stamatevalentin125/Cloud_Project/1.0.0

Bibliography

- https://radixweb.com/blog/guide-to-food-delivery-app-development
- https://www.spaceotechnologies.com/blog/technologies-to-develop-food-delivery-apps/
- https://www.grandviewresearch.com/industry-analysis/online-food-delivery-services-market